

REMARKS**I. INTRODUCTION**

Applicant respectfully requests reconsideration of all pending claims in view of the amendments and remarks presented herein. This is the fourth substantive response that applicant has prepared and submitted to the Examiner. In each instance, applicant has commented extensively on the language of the claims that provides patentable distinctions over the cited prior art. Further, applicant has repeatedly offered to discuss and/or demonstrate the present invention in view of the examiner's rejections. However, the Examiner has maintained his rejection of most of the claims notwithstanding applicant's arguments by citing new art. In each instance, the newly cited art has failed to supply missing features. Applicant believes that the grounds of rejection have been adequately traversed yet again, and that the claims are in condition for allowance.

In sum, applicant submits that the Examiner has failed to make a prima facie case that the pending claims are not patentable over the cited references. The cited references do not, either alone or in combination, teach or suggest the invention as claimed. Most significantly, the Examiner has not provided any satisfactory explanation of how the cited art would provide server side editing features for a dynamic web application that permit the application to look similar and function like the normal running application even during editing.

A. Status of Claims

Claims 1-8, 22-33, 41-43, 51-96 and 114-127 were examined and remain pending. Claims 9-21, 34-40, 44-50 and 97-113 have been previously cancelled as drawn to non-elected subject matter. Claims 2, 5, 26-31, 52, 54-55, 59, 61, 64, 66, 68, 70, 73-74, 76, 78, 80-88, 90, 92-94, 114, 116, 118, 121-123 and 125 are amended herein for increased clarity.

The Examiner has continued to reject claims on the basis of prior art, as follows:

- (1) claims 26, 30, 32-33, 66-68 and 71-72 were rejected under Section 102(e) as being anticipated by U.S. Patent No. 6,151,609 to Truong;
- (2) claims 1, 22, 90-96 and 114-119 and 121-127 were rejected under Section 103(a) as being obvious in view of the combination of Truong and U.S. Patent No. 6,529,910 to Fleskes;

- (3) claims 2-8, 23-25, 31, 51-55, 59-65, 69, 74, 76, 82, 84-86 and 88 were rejected under Section 103(a) as being obvious in view of the combination of Truong, Fleskes and U.S. Patent No. 5,987,513 to Prithvirag et al.;
- (4) claims 31-33, 41-43, 51-55, 59-72, 74-77, 82, 84-86 and 88 were rejected under Section 103(a) as being obvious in view of the combination of Truong and Prithvirag; and
- (5) claims 27-29 were rejected under Section 103(a) as being obvious in view of the combination of Truong and U.S. Patent No. 6,452,609 to Katinsky;

Applicant notes that the following claims stand rejected on two different bases, as set forth in items 3 and 4 above: claims 31, 51-55, 59-65, 74, 82, 84-86 and 88.

The Examiner did indicate that claims 56-58, 73, 78-81, 83, 87, 89 and 120 contain allowable subject matter. Applicant appreciates the indication of allowability as to these dependent claims. The Examiner has apparently withdrawn the previous indication of allowability of claims 27-29 on the basis of newly cited art to Katinsky, as indicated in item 5 above.

Applicant respectfully traverses the prior art rejections as discussed below. Further, applicant has submitted herewith a Declaration Under Rule 131 to remove Fleskes and Katinsky as prior art references. Applicant and his patent counsel remain willing and available to interview with the Examiner and/or his supervisor as necessary to discuss the prior art rejections, the relevant claim language, and applicant's arguments traversing the rejections, and to arrange for a demonstration of applicant's system if necessary.

B. Summary of Arguments Traversing the Prior Art Rejections

Applicant has developed a unique editor for server based dynamic web applications. Advantageously, when editing a dynamic web application using applicant's editor, the application looks similar to and functions like the normal running application as viewed by the end user. Achieving this result is a much more difficult task than simply editing static pages. Execution of dynamic web applications transforms page templates into generated pages for display to the user. Editing the application means *changing page templates*. Displaying the application to look like and function like the end user's view requires that the editor *show the generated pages*. So, in order to edit dynamic web applications, the editor must be capable of displaying the generated pages while at the same time modifying the page templates.

Applicant's editor does so by running the application during editing and showing the generated pages. Upon a modification, a special change request is created and sent to the server to perform an appropriate change to the page template.

In rejecting the claims, the examiner relies primarily on the editor disclosed in the Truong patent. However, Truong's editor just shows the source code of the application during editing, which looks very different to the end users' view. The examiner acknowledges that "Truong doesn't explicitly disclose an editor capable of directly operating on the documents displayed by the browser thereby allowing the user to work on a functional application during development." The examiner then cites Fleskes as disclosing this feature, and alleges that it would have been obvious to combine Truong and Fleskes. (Paper No. 18 at pp. 4-5). However, applicant respectfully disagrees that Fleskes discloses this feature, and submits that the combination is improper and that the examiner has failed to make a *prima facie* case for obviousness, as discussed below. Further, applicant has submitted a Declaration Under Rule 131 to remove Fleskes as a prior art reference and requests that any rejection based on Fleskes be withdrawn.

Applicants editor contains specific functions to add, modify, and delete components, and it specifically can handle components that generate browser code and/or operate on the server side. It is much more difficult to correctly display a server side component during editing than a browser built-in, because a server-side component must be executed on the server in order to generate the HTML code needed to display the component.

Truong's editor, just like any conventional text editor, has operations to add, modify and delete characters, but not components. The examiner refers to the prior art section of Truong, wherein Truong explains browser built-in components. Truong's editor implementation uses these components, but Truong's edit operations do not operate on the components. In addition, browser built-in components are clearly different from server-side components e.g. they do not generate browser code.

II. DISCUSSION OF THE CITED ART

A. Truong

The Truong patent was discussed in detail in applicant's Amendment and Remarks in Response to Office Action dated February 28, 2003, which is incorporated herein by reference. In sum, Truong discloses a web-based text editor which does not permit real time editing of fully

functional, running web applications, as the examiner has now acknowledged. (*See* Paper No. 18 at pp. 4-5).

B. Combination of Truong and Fleskes

The examiner now acknowledges that “Truong doesn’t explicitly disclose an editor capable of directly operating on the documents displayed by the browser thereby allowing the user to work on a functional application during development.” (Paper No. 18 at pp. 4-5). However, the examiner alleges that Fleskes discloses this feature, and that it would have been obvious to combine Fleskes with Truong since “editing pages dynamically while page is being executed or running makes editing or modifying pages more visual.” (*Id.* at p. 5). Applicant respectfully disagrees, and submits instead that the examiner has failed to make a *prima facie* case for obviousness. First, Fleskes should be removed as a prior art reference based on applicant’s Rule 131 Declaration submitted concurrently herewith. Second, Fleskes does not disclose this feature. Third, there is no suggestion or motivation to combine the references, there is no reasonable expectation of success, and the proposed combination does not teach or suggest all of the claim limitations. The combination is improper and should be withdrawn.

Applicant submits that Fleskes’ editor is not running inside the web browser. Figure 1 and 2 of Fleskes do not show the editor as residing inside the browser, and there is no teaching or suggestion anywhere in Fleskes that it would do so. In addition, the fact that Fleskes explicitly describes a GUI editor indicates that the editor has a GUI interface and not a web user interface. In contrast, applicant’s editor is completely browser based, and this distinction is expressly reflected in applicant’s claim language. Also, there is no teaching or suggestion in Fleskes that its editor displays a running application. In fact, to the contrary, Fleskes explicitly describes editing images, and it is well known that images do not contain scripts that could make them executable. In contrast, applicant’s editor edits web pages possibly containing scripts, wherein the scripts stay functional during editing. This distinction is also expressly reflected in applicant’s claim language.

Furthermore, the combination of Truong and Fleskes does not make sense because images use very different data formats than text. On the other hand, Truong’s editor works for text files only. (*See* Truong at column 10, lines 45-50 “*providing the text of the selected file to the web browser for editing as shown in figure 5 ... the text may be edited at the web browser*”). Applicant therefore submits that Truong’s text editor cannot reasonably be modified by

incorporating features from Fleskes' editor, and in any event, doing so would not yield an editor capable of operating directly on a document displayed by the browser.

Applicant's editor is used to edit web application programs, i.e. page templates. It does so by showing generated pages to the developer (user) who is using the editor, with the purpose of editing the application itself. In contrast, Fleskes editor edits the images, but not the program that produced the images. In column 5 lines 54-64, Fleskes states that the editor can edit images generated by programs making explicit that the editor edits the image and not the program.

Applicant submits that Fleskes' editor can edit images, but not pages and consequently not generated pages. In addition, the combination of Truong with a page editor on the client side for editing generated pages would result in a useless mechanism. Truong states that upon a save operation, the complete file is transferred from the client to the server, replacing the file on the server. Using the mechanism created by the examiner's proposed combination, an edited version of a generated page would overwrite the corresponding page template, which would make the page template dysfunctional. Therefore, the combination of Truong with Fleskes is improper.

Applicant submits that Fleskes' editor can edit images, but not pages, and consequently also not generated pages. In addition, editing of page templates does not cause the display of a web application in working order with an appearance similar to the generated pages normally displayed to the end user. In general, the generated pages look different from the page templates because, for example, they contain additional portions generated during execution of the application. In addition, page templates do not appear in working order because they are lacking the generated parts. In contrast, applicant's editor displays the applications in working order with an appearance similar to the end user's view. This distinction is expressly reflected in applicant's claim language.

C. Combination of Truong, Fleskes and Prithvirag

The Prithvirag patent was discussed in detail in applicant's Amendment and Remarks in Response to Office Action dated February 28, 2003, which is incorporated herein by reference. In sum, Prithvirag is cited as disclosing page templates. However, because neither Truong nor Fleskes teaches or suggests the editor recited in applicant's claims, as discussed above, the addition of page templates from Prithvirag does not suffice to supply all of the missing features from applicant's claims. In addition, Fleskes should not be considered as prior art, as noted in applicant's Rule 131 declaration.

D. Combination of Truong and Prithvirag

See discussion in Sections B and C above.

E. Combination of Truong and Katinsky

Katinsky should not be considered as a prior art reference, as discussed in applicant's Rule 131 declaration submitted herewith.

III. DISCUSSION OF THE REJECTED CLAIMS**A. OVERVIEW**

Applicant submits that his claim language adequately recites key differences allowing the claims to distinguish over the cited art.

Advantageously, when editing a dynamic web application using applicant's editor, the application looks similar to and functions like the normal running application as viewed by the end user. In contrast, Truong's editor just shows the source code of an application, and Fleskes' editor works on images, not applications. However, as is well known, images can not be executed. This distinction is explicitly reflected in applicant's claim language. For example, claim 1 recites "*an editor operating directly on the pages displayed by the browser via the editing features, thereby allowing the user to work on a functional application during development;*" claim 26 recites "*at least a part of the second document appears and functions similar to the first document;*" claim 59 recites "*documents which look and function similar to the display documents;*" and claim 90 recites "*scripts contained in said document remain functional.*"

Applicant's editor is capable of performing editing functions operating on components. In contrast, Truong's editor functions operate on characters. This distinction is explicitly reflected in applicant's claim language. For example, claim 22 recites "*an editor operable within the web browser for inserting, deleting, and modifying components on document templates;*" claim 51 recites "*A system for editing components;*" claim 74 recites "*an editor capable of performing edit functions maintaining components on document templates.*"

Applicant's components generate HTML code *before* the document is actually transmitted to and shown inside the browser. Browser built-in components as referenced by Truong directly operate inside the browser without HTML generation and are executed during display of a document. This distinction is also reflected in applicant's claim language. For

example, claim 6 recites “*the server computer further comprising a plurality of components*”; claim 22 recites “*a document generator for processing document templates, executing components and for generating documents from the document templates that are understandable by the web browser*”; claim 51 recites “*a plurality of components encapsulating browser code*”; claim 74 recites “*a plurality of components, that include instructions to generate browser code*”; claim 114 recites “*components for execution on the server, at least one of the components including . . . second program instructions to generate browser code*”; and claim 125 recites “*running the application, thereby executing components and generating a generated document*.”

B. CLAIMS 1-8, 22-25, 76, 90-96, 114-119, 121-127

Claim 1

Claim 1 stands rejected based on the combination of Truong and Fleskes. However, applicant respectfully traverses the rejection. Applicant has submitted a Declaration Under Rules 131 to remove Fleskes as a prior art reference. On that basis, the combination is improper.

Claim 1 is an independent claim directed to a “*software development system*” for an online data network, wherein the user (client) computer runs a browser program to interpret web pages posted by the server. The system includes “*a page generator*” and “*an editor*,” whereby the page generator runs the application being developed during editing. This allows the developer to see application pages during editing that look and function similar to the end users’ view of the application. In contrast, Truong’s editor does not follow the WYSIWYG principle, but instead shows the source code of an application during editing. This can be seen from Truong’s Figure 5. The examiner has acknowledged that “Truong doesn’t explicitly disclose an editor capable of directly operating on the documents displayed by the browser thereby allowing the user to work on a functional application during development.” (Paper No. 18 at pp. 4-5) However, the examiner reasons that it would however been possible to combine Truong with Fleskes. For all the reasons discussed in section IIB above, the combination Fleskes and Truong is improper.

The examiner cited Truong column 10 line 45-50 as disclosing a page generator running an application being developed and sending generated documents to the browser. Applicant respectfully disagrees with the examiner’s reading of the cited portion, and submits that instead, the cited portion discloses “providing the text of the selected file to the web browser for editing.”

In applicant's reading of this cited portion, it is clear that the text of the file is sent to the browser without execution, i.e. *without running the application being developed*. This can also be seen from Truong's Figure 5 that shows Truong's editor editing a script, whereby the editor displays the source code and not the generated page. In contrast, the claim explicitly requires running the application being developed.

The examiner did not propose the combination with Fleskes for this part of the claim. Nevertheless for the second part of the claim the examiner cites Fleskes at column 22, lines 25 to 27. This portion discusses a generation process, but makes clear that this happens only at runtime, when a page is rendered to the end-user.

Applicant submits that adding a step executing the application during editing to Truong would make Truong's editor dysfunctional because of the following: As described by Truong in column 10, lines 55-58, after editing, the edited text is sent to server and saved in the edited file. If Truong would have included a step of executing the application, this save operation would save also changes made by running the application into the page template file, not just the changes the user did by editing. Then this would overwrite the application by its output and consequently render it dysfunctional.

The examiner cited Fleskes column 5 line 54 to 64 and column 22 line 25 to 27 as disclosing an editor capable of directly operating on the documents displayed by the browser thereby allowing the user to work on a functional application during development. However in applicants reading the cited portion discloses a GUI editor for images, which works separately from the browser. So Fleskes editor edits images while applicants claim requires the editor to work on applications. In addition Fleskes editor has a GUI interface and is not part of the browser and so can not directly operate on the documents displayed in the browser as required by the claim.

Applicant previously amended the claim requiring the editor to operate via the editing features. The examiner did not include reasoning towards this point.

For all the foregoing reasons, applicant submits that claim 1 as pending is patentable over the cited combination.

Claim 6

Claim 6 stands rejected based on the combination of Truong, Fleskes and Prithvirag. However, applicant respectfully traverses the rejection.

Claim 6 is an independent claim directed to a software development system that allows the user to create client server based applications by plugging together components. Applicant's invention provides components that can interactively communicate with the user on the client computer and that can execute instructions on the server as well. Claim 6 as pending explicitly requires components that operate on the server, reciting "at least one component that reacts interactively on user input by executing instructions contained in said component on the server." Applicant previously amended the claim adding "contained in said component". The examiner's comments on page 8 of the office action do not appear to address this amendment. Instead, the examiner cited Truong at column 2, lines 1 to 5, as disclosing a plurality of components residing in the data store on the server including components that react interactively on user input by executing instructions on the server. However, Truong refers to browser built in HTML forms and HTML fields. Since these components are built into the browser, they reside in the data store of the client and they are executed on the client, not the server. HTML forms and fields can send data to the server, but they do not contain instructions for execution on the server. This should be clear since the complete browser with everything in it runs on the client only. In contrast the previously amended claim clearly requires components to contain instructions on the server.

The examiner also cites Truong at column 5, lines 60 to 65 as disclosing components. However, this cited portion explains how internet servers send URLs and pages to each other and then finally to a client. The cited portion does not teach or suggest components as recited by applicant

The examiner also cites Truong Figure 3c item 128 as disclosing a data store. The cited portion states "*store each string as a variable,*" which indicates that data is stored but not the fact that the data store contains components.

Claim 22

Claim 22 stands rejected based on the combination of Truong and Fleskes. However, applicant respectfully traverses the rejection. Applicant has submitted a Declaration Under Rules 131 to remove Fleskes as a prior art reference. On that basis, the combination is improper.

The examiner cited Trung column 11 line 17-20 as disclosing an editor operable with the web browser for inserting, deleting, and modifying components on document templates. In applicants reading however the cited portion discloses that the WINDOWS editing commands such as copy, insert, delete, and paste may be used, if the browser is running on the WINDOWS operating system. The cited portion does not seem to disclose that the editing commands work on components and it does not seem to disclose a modify operation as required by the claim. In contrast at column 10 line 47 Truong writes “the text may be edited at the browser using editing features such as delete, select, search, copy, paste, and the like” In applicants reading this makes clear that Truong's editing features work on text and not on components.

The examiner cited Truong column 10 line 45 to 50 as disclosing a document generator for processing document templates, executing components and for generating documents from the document templates that are understandable by the web browser. In applicants reading the cited portion discloses providing the text of the selected file to the web browser. The cited portion does not seem to disclose components nor a document generator for executing components as required by the claim.

Claim 51

Claim 51 stands rejected based on the combination of Truong, Fleskes and Prithvirag, or on the combination of Truong and Prithvirag, based on the same reasoning as applied to claim 6. (Paper No. 18 at p. 9) However, applicant respectfully traverses the rejection.

Claim 51 is an independent claim describing a system for editing components on web document templates. The main distinction between Truong's editor and the claimed invention is that Truong does not have components and his editor shows source text only. This distinction is represented in the claim language, for example “*a user interface for editing functions used for maintaining components on document templates.*” The cited references do not teach or suggest a user interface having editing functions for maintaining components. The editing functions actually disclosed in Truong are clearly identified as working on text by stating: “*the text may be edited at the web browser using editing features.*” (See Truong at column 10, lines 47 to 50.) In addition, Figure 5 of Truong shows the user interface of the editor, but does not show “a user interface for editing functions used for maintaining components on document templates,” as recited in claim 51

Claim 51 was previously amended to explicitly requires that the components encapsulate browser code.

In his reasoning with regard to claim 6, the examiner cited Truong as disclosing a plurality of components. However, as discussed above, Truong discloses browser built in HTML forms and HTML fields. Since these components are built into the browser, they do not encapsulate or generate browser code as required by the claim.

Claim 59

Claim 59 stands rejected based on the combination of Truong, Fleskes and Prithvirag, based on the same reasoning as applied to claim 6. (Paper No. 18 at p. 10.) However, applicant respectfully traverses the rejection.

Claim 59 is an independent claim directed to a software development system for dynamic web documents capable of displaying functional applications during editing. Applicant submits that claim 59 is quite different than claim 6 because it is specifically concerned with editing technology, and the reasoning for claim 6 is wholly inapplicable to claim 59.

As explained above, Truong's editor works on the source code of an application and consequently does not follow the WYSIWYG principle and does not execute the application during editing. In contrast, the inventive editor is capable of executing the application being developed during editing and so the application appears functional during editing.

This distinction is expressed by the following claim language, "*the editor program comprising first instructions for requesting the document generator to process a dynamic web document leading to a generated document.*" This language makes clear that the editor indeed executes the application during editing. Additional claim language, namely "*instructions to modify the dynamic web document*" make clear that the dynamic web document that is being edited is the one being requested. Applicant previously amended the claim to clarify that the generated document looks and functions similar to the display document which is the users view on the dynamic web document.

Claim 74

Claim 74 is an independent claim describing a software development system using components on web document templates. The examiner rejected this claim using his reasoning

towards claim 6. Applicant refers to his discussing of claim 6 above. Truong's editor works on text only while applicant's editor contains specific editing functions for handling components. This distinction is represented in the claim language, namely "*an editor capable of performing edit functions maintaining components on document templates.*" None of the cited references disclose that Truong's editor is capable of performing edit functions maintaining components. The editing functions disclosed in Truong column 10 lines 47 to 50 clearly identify themselves as working on text "*the text may be edited at the web browser using editing features*".

In addition claim 74 explicitly requires components to generate browser code using the claim language "a plurality of components, that include instructions to generate browser code for transmission to the first software program". In his reasoning with respect to claim 6 the examiner cited Truong column 2 lines 1 to 5 as disclosing a plurality of components. However, Truong refers to browser built in HTML forms and HTML fields. Browser built in components however do not generate browser code for transmission to the browser program as required by claim 74.

Claim 90

Claim 90 is an independent claim. The examiner did include reasoning about claim 90 twice, once together with the reasoning of claim 1 and a second time independently. Both reasonings seem to contradict each other, because the first reasoning relies on Fleskes while the second one does not. Because the second reasoning is a literal copy of the previous reasoning, also not taking into account applicants recent amendments applicant assumes that the second reasoning was copied into the document by error.

In his reasoning towards claim 1 the examiner proposes to use Fleskes' editor. The examiner did not give any reasoning that Fleskes editor could keep scripts running during editing. According to Fleskes column 5 line 54 – 64 Fleskes editor is for images only. Images, however, do not contain scripts that could stay functional during editing. Therefore applicant submits that Truong combined with Fleskes do not disclose an editor allowing editing of documents "whereby scripts contained in said document remain functional during editing" as required by the claim.

The examiner did also not give reasoning for a first software program for execution within the browser. In contrast according to Fleskes column 5 line 54 – 64 Fleskes editor is GUI based.

Claim 114

Claim 114 is an independent claim. The examiner did not include specific reasoning for this claim, but used the same reasoning as for claim 1. Reasoning however seems not to be appropriate, because Claim 114 explicitly claims components that include features to cooperate with the editor. In contrast Claim 1 and all its reasoning is not involved with components at all.

In his reasoning towards claim 6 the examiner discusses components, however Truong discloses only browser built-in components, which can not generate browser code. In contrast the claim requires “components including second program instructions to generate browser code”. In addition browser built-in components run on the client as part of the browser. In contrast the claim requires “a plurality of components for execution on the server”.

Claim 125

Claim 125 is an independent claim directed to a method for editing an application. The examiner did not include specific reasoning for this claim, but used the same reasoning as for claim 1. In the discussion towards claim 1 applicant reasoned that neither Truong nor Fleskes suggest running an application during editing. In addition running an application during editing would make Truong dysfunctional. In contrast the claimed method for editing an application includes a step of running the application.

In addition claim 125 contains limitations towards components. The reasoning towards claim 1 seems to be inadequate for this, because Claim 1 and all its reasoning is not involved with components at all.

In his reasoning towards claim 6 the examiner discusses components, however Truong discloses only browser built-in components, which are executed during document display. In contrast claim 125 requires components to be executed during document generation prior to document display using two steps, a step of “running the application, thereby executing components and generating a generated document” and a second step of “displaying a view of the generated document”.

C. CLAIMS 51-55, 59-65, 69, 74, 76, 82, 84-86, 88

The examiner listed these claims twice as being rejected, once in his section 7 in view of Fleskes and once in section 8 without citing Fleskes. The reasoning seems to be identical in both sections. As far as the independent claims 51, 59, and 74 are concerned, the examiner's reasoning refers to the reasoning of claim 6, which is discussed above.

D. CLAIMS 41-43, 75,77

The examiner rejected these claims without giving any explicit reasoning. Applicant therefore submits that the examiner has failed to make a prima facie case.

E. CLAIM 26

The examiner cited Truong at column 10 lines 45-50 as disclosing features which permit editing of the first document such that at least part of the second document appears and functions similar to the first document. In applicant's reading of the cited text, Truong talks about functions for editing source text. Source text, however, does not appear similar to the normal view of a document and it does not function at all because it is not executed. This is also shown by Truong's figure 5. In addition, in his reasoning with respect to claim 1, the examiner states that Truong does not explicitly disclose an editor capable of directly operating on the documents displayed by the browser thereby allowing the user to work on a functional application during development. If the application is not functional at all during development then it certainly does not function similar to the first document. In contrast, applicant's claim requires that "at least a part of the second document appears and functions similar to the first document."

F. DEPENDENT CLAIMS

All of the remaining rejected dependent claims are patentable for the reasons stated with regard to the corresponding independent claim.

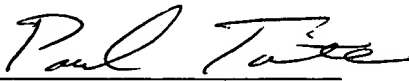
IV. CONCLUSION

For all the foregoing reasons, applicant submits that the claims are in condition for allowance, and the examiner's favorable reconsideration to that end is solicited. If additional questions remain, the examiner is encouraged to telephone the undersigned.

Respectfully submitted,

DERGOSITS & NOAH LLP

Dated: April 27, 2004

By: 
Paul K. Tomita
Reg. No. 43,196

Four Embarcadero Center, Suite 1450
San Francisco, California 94111
(415) 705-6377 tel
(415) 705-6383 fax
rnebb@dergnoah.com

APPENDIX

1. (previously amended) A software development system for applications that run on a data network which couples a server computer and a client computer, wherein the client computer runs a browser program, comprising

a page generator running an application being developed and sending generated documents to the browser for display as pages including additional editing features for interpretation by the browser program;

an editor directly operating on the pages displayed by the browser via the editing features, thereby allowing the user to work on a functional application during development.

2. (currently amended) A software development system as claimed in claim 1, further comprising a plurality of components, and wherein developed applications comprise at least one page template capable of containing components, and wherein the editor provides features to insert, modify and delete a component ~~components-on~~ at least one page templates, and wherein the page generator executes selected ~~the~~ components on page templates.

3. (previously amended) A software development system as claimed in claim 2, wherein at least one of the components reacts interactively on user input by executing instructions of said component on the server.

4. (original) A software development system as in claim 3, wherein at least one of the components contains at least one other component.

5. (currently amended) A software development system as in claim 3, wherein the set of components on pages generated from ~~a single~~ at least one page template can vary for different requests of ~~the same~~ said page template.

6. (previously amended) A software development system for use in a data network which couples a server computer to a client computer, wherein the client computer includes a first software program for generating a request for one or more pages

APPENDIX

from the server computer and for displaying pages, and wherein the server computer includes a second software program for receiving and processing the request from the client computer, for generating and storing pages, and for transmitting pages to the client computer in response to requests, the server computer further comprising:

- a data store,

- a plurality of components residing in the data store, including at least one component that reacts interactively on user input by executing instructions contained in said component on the server;

- a plurality of page templates residing in the data store, at least one page template having at least one selected component incorporated therein; and

- a server processor controlled by a third software program, said program providing instructions for selecting a page template based on the request from the client computer and instructions for generating a page from the page template for transmission to the client computer.

7. (original) The development system of claim 6, further comprising a component editor controlled by a fourth software program, said program providing instructions for interactively editing selected components on a page template.

8. (original) The development system of claim 6, wherein a component is nested within a component.

9. (previously amended) A method for generating documents for display by a browser using components that react interactively on user input by executing instructions on a server, comprising the following steps for execution on the server upon a document request:

- assigning a unique identifier to at least one of the components; and
- embedding the unique identifier into a generated document.

10. (previously amended) The method of claim 9, further comprising storing data on the server representing at least one of the components.

APPENDIX

11. (previously amended) The method of claim 10, further comprising:
analyzing the request sent by the browser for unique identifiers; and
calling a function for the interactive components referenced by at least one of the
unique identifiers contained in the request.
12. (previously amended) The method of claim 11, wherein at least one of the components
is contained on a document template.
13. (original) The method of claim 11, wherein at least one of the components is called by a
program.
14. (original) The method of claim 11, wherein at least one of the components is called by a
another component.
15. (original) The method of claim 11, wherein the data is stored into an object of an object
oriented programming language and wherein the function is a method of the object.
16. (original) A method for implementing client server applications, comprising:
storing data objects on a server and assigning a unique identifier to each data
object;
dynamically generating a document with the unique identifier embedded in the
document; and
analyzing requests for unique identifiers and calling at least one function for a
data object associated with one of the unique identifiers found in the request.
17. (original) The method of claim 16, wherein the unique identifier is embedded
inside a uniform resource locator contained in a tag of the document.
18. (original) The method of claim 16, wherein the unique identifier is embedded
in scripts contained in the document.

APPENDIX

19. (original) The method of claim 16, wherein the unique identifier is unique within a single session.

20. (previously amended) The method of claim 16, wherein the unique identifier is unique within all documents generated by a single server within a defined time period.

21. (original) The method of claim 16, wherein the data objects are created by an object-oriented programming language and said function is a method of one of these objects.

22. (previously amended) A computer running an application to develop and maintain applications using a web browser, comprising:

an editor operable within the web browser for inserting, deleting, and modifying components on document templates; and

a document generator for processing document templates, executing components and for generating documents from the document templates that are understandable by the web browser.

23. (previously amended) A computer as in claim 22, wherein the editor operates functional applications in an edit mode permitting editing directly in the web browser.

24. (previously amended) A computer as in claim 23 wherein at least one of the components contains instructions and can react on subsequent document requests containing user responses by executing selected instructions of said component.

25. (previously amended) A computer as in claim 24, wherein the computer further comprises:

a store of component classes, each component class implementing one component kind; and

a parser able to detect components marked on document templates;

APPENDIX

wherein the document generator works upon a document request using component classes to generate browser code; and

wherein the editor is capable of showing a menu of components for insertion into the document templates.

26. (currently amended) A system to modify documents on a server in a data network which couples said server computer to a client computer, the server computer comprising:

a document store;

a first software program including instructions for transforming at least one first document retrieved from the document store into a second document having features which permit editing of the first document such that at least a part of the second document appears and functions similar to the first document; and

a second software program including instructions to receive information from the client computer and instructions to modify the first documents stored in the document store.

27. (currently amended) The system of claim 26, wherein the first document includes at least one component being executed by the first software program ~~and wherein the second document includes at least one handle to indicate the position of the component to the user.~~

28. (currently amended) The system of claim 27, wherein the second document includes handles and choosing one of the handles selects a component for an editing operation.

29. (currently amended) The system of claim 28, wherein at least one handle indicates the position of at least one component contained in the first document and said editing operation ~~is chosen from the group of~~ includes modifying the component, deleting the component, and displaying information regarding the component, ~~and inserting a new component.~~

APPENDIX

30. (currently amended) The system of claim 26, wherein the features ~~are~~ include scripts.

31. (currently amended) The system of claim 30, wherein the scripts ~~are generated specifically for the second document and encapsulate information which is incorporated into~~ from the first document.

32. (original) The system as in claim 26, wherein the features incorporate information regarding the first document into the second document.

33. (previously amended) A system as in claim 32, wherein the information incorporated into the second document is used on the client computer in order to send change requests for the first document to the server.

34. (previously amended) A method for generating a document for display in a browser from a document template containing components, comprising:

for each component denoted on the document template, identifying a component class of the component; and

based on data contained in a request initiated by the browser storing a first object of the component class, the first object representing the component.

35. (previously amended) The method of claim 44, further comprising calling a method of said component class to generate browser code, said method being the constructor.

36. (original) The method of claim 34, further comprising, for all components having a name attribute, looking up the component object in session memory based on said name attribute.

APPENDIX

37. (previously amended) The method of claim 34, further comprising, for at least one component kind, for all components denoted on the document template having said component kind;

generating a unique identifier;

assigning said unique identifier to said object, and

embedding said unique identifier into the browser code.

38. (previously amended) The method of claim 37, further comprising:
inserting objects for the components of at least said component kind into a list of listening components;

working through all objects stored in the list of listening components whose unique identifier occurs inside a name in the form data set; and

calling a method of at least one of these objects.

39. (previously amended) The method of claim 34, wherein the document template is parsed into a list of nodes, including text and component nodes, said method further comprising:

determining if the current node is text or a component;

if component, then calling a method for the component, comprising:

evaluating the attributes of the component if necessary;

identifying the component class associated with the component; and

calling the constructor method of the component class,

said constructor method generating browser code;

if text, then generating the text; and

repeating these steps for each node.

40. (original) The method of claim 39, wherein at least one component contains nested components and the method of claim 39 is recursively performed for all nodes nested inside the component.

41. (previously added) A software development system as in claim 1, the editor comprising a client part for execution on the client computer.

APPENDIX

42. (previously added) A software development system as in claim 41, wherein the client part comprises instructions that are automatically downloaded from the server prior to editing.

43. (previously added) A system as in claim 26, additionally comprising at least one script for automatic download to the client that works in cooperation with the second document to permit editing of the first document.

44. (previously amended) The method of claim 34 wherein storing the first object comprises creating a new object as necessary.

45. (previously amended) The method as in claim 34 wherein components are denoted on document templates using tag syntax.

46. (previously added) The method as in claim 45 wherein the tag name identifies a component class.

47. (previously amended) The method as in claim 36 wherein components are denoted on document templates using tag syntax, wherein the tag name identifies a component class.

48. (previously added) The method as in claim 36 wherein the component object, if found, is reused to store the first object.

49. (previously added) The method as in claim 36 wherein in case a component has a name attribute but no component object is found a new object is created and stored under said name in session memory.

50. (previously added) The method as in claim 49 wherein new objects are created for all components not having a name attribute.

APPENDIX

51. (previously amended) A system for editing components on web document templates for use with a first software program including first instructions for generating a document request to obtain at least one generated document from a second software program and for displaying the generated document, the second software program capable of receiving and processing the document request and of transmitting first documents to the first software program in response to requests, said system comprising:

a plurality of components encapsulating browser code,

a plurality of document templates,

the second software program transmitting, while processing selected requests, second documents to the first software program that make the first software program display a user interface for editing functions used for maintaining components on document templates,

a third software program used by the second software program while processing selected document requests, the third software program including third instructions for modifying document templates in order to perform said editing functions.

52. (currently amended) The system of claim 51, wherein at least some components include fourth program instructions including steps to generate browser code for transmission to the first software program.

53. (previously added) A system in claim 52 running on a data network, coupling a server computer and a client computer, the first program running on the client computer, the second program running on the server computer.

54. (currently amended) A system in claim 52 wherein second documents ~~are~~ include HTML pages with embedded scripts.

55. (currently amended) The system of claim 52, wherein the edit function includes adding a component to a document template, removing a component from a document template, and modifying ~~attributes of~~ a component on a document template.

APPENDIX

56. (previously added) The system of claim 52, further comprising a fifth software program used by the second software program while processing selected document requests, the fifth software program including fifth instructions for generating generated documents from document templates thereby calling fourth program instructions.

57. (previously added) The system of claim 56, wherein the generated document includes, if requested in edit mode, edit features for interpretation by the first software program.

58. (previously amended) The system of claim 56 further comprising instructions to allow the user to click on the generated document to select items to perform edit functions on.

59. (currently amended) A software development system for developing dynamic web documents ~~for transformation into display documents for display to a user,~~ comprising:

an editor program for editing dynamic web documents,

a document generator for generating generated documents from dynamic web documents which look and function similar to the end user's view of the display documents,

the editor program comprising first instructions for requesting the document generator to process a dynamic web document leading to a generated document,

the ~~editor program~~ system further comprising second instructions for displaying at least some information items contained on said generated document in a view which allows the user to select an item to which a modification function will be applied,

the editor program further comprising third instructions to modify the dynamic web document to perform said modification function.

APPENDIX

60. (previously added) The software development system of Claim 59 running on a data network, which couples a server computer and a client computer, the document generator running on the server computer the editor at least partly running on the client computer.

61. (currently amended) The software development system of claim 60, ~~wherein the document generator~~ further comprising fourth instructions for execution during document generation to collect edit-information for use by the editor.

62. (previously added) The software development system of claim 60, wherein the editor uses a web browser for displaying said view.

63. (previously added) The software development system of claim 60, able to automatically repeat requesting the document generator to process the dynamic web document if required.

64. (currently amended) The software development system of Claim 59 further comprising a plurality of components including at least one component marked on said the dynamic web document, and ~~components~~ including instructions for use by the document generator to generate browser code.

65. (previously added) The software development system of claim 64, wherein the editor uses a web browser for displaying said view.

66. (currently amended) The software development system of claim 64, wherein modification functions include insert of a component, delete of a component, and modify ~~attributes of~~ a component.

67. (previously added) The software development system of claim 59, wherein said view looks, except for editing features, similar to the end-user view of the generated document.

APPENDIX

68. (currently amended) The software development system of claim 59, ~~wherein the document generator further comprises~~ sixth instructions to collect edit-information for use by the editor, said sixth instructions for execution during document generation.

69. (previously added) The software development system of claim 68, wherein the editor uses the edit-information to correctly modify the dynamic web document.

70. (currently amended) The software development system of claim 69, further comprising a plurality of components ~~marked on the dynamic web document~~, wherein the edit-information comprises position information on ~~the~~ selected components ~~contained in marked on the document template~~.dynamic web document.

71. (previously added) The software development system of claim 59, wherein the editor uses a web browser for displaying said view.

72. (previously added) The software development system of claim 71, wherein first instructions comprise seventh instructions for initiating a reload in the browser.

73. (currently amended) The software development system of claim 59 the editor program further comprising eighth instructions to display information on at least one element of at least one ~~the~~ dynamic web document, that is replaced during document generation, without requesting the document generator to regenerate the generated document.

74. (currently amended) A software development system for document templates that are intended for transformation into generated documents for display by a first software program, the first software program including first instructions for generating a document request to obtain at least one generated document and for displaying the generated document, comprising:

APPENDIX

a plurality of components, that include instructions to generate browser code for transmission to the first software program,

an editor capable of performing edit functions maintaining components on document templates, the components capable to cooperate with the editor,

a plurality of document templates having components denoted thereon, and

a document generator comprising second instructions to, upon a document request, generate generated documents from at least one document template for display by the first software program wherein the set of components on the generated document can vary for different document requests for ~~the same~~ said document template.

75. (previously added) The software development system as in claim 74, wherein edit function comprises adding a component, modifying a component, and deleting a component.

76. (currently amended) The software development system as in claim 74, wherein tag syntax is used to denote at least one components on at least one document templates, whereby the tag name identifies the component kind.

77. (previously added) The software development system of claim 74 running on a data network, which couples a server computer and a client computer, the document generator running on the server computer the editor running, at least partly, on the client computer.

78. (currently amended) The software development system as in claim 74, wherein at least one component, that can react interactively on subsequent document requests, can be excluded from the generated document.

79. (previously added) The software development system as in claim 78 further comprising third instructions to prevent excluded components from reacting on subsequent document requests.

APPENDIX

80. (currently amended) A software development system as in claim 79, said third instructions comprising fourth instructions to, upon a first document request, store information in session memory on ~~all~~ some of the components, that are present on the generated document, and fifth instructions to, upon subsequent document requests, only react on components that have been remembered in session memory thereby avoiding tampering with excluded components on the side of the first program.

81. (currently amended) A software development system as in claim 74 wherein at least one ~~document template has a first and a second component denoted thereon in a way that the first component contains the second component, the first component contains~~ sixth instructions to decide about exclusion of the second components nested inside the first component from the generated document.

82. (currently amended) A software development system as in claim 74 the ~~editor system~~ able to provide an editable view taking the varying set of components into account.

83. (currently amended) A software development system as in claim 74 the ~~editor system~~ able to provide an editable view that includes and excludes selected components similarly as the final application.

84. (currently amended) A software development system as in claim 74 wherein the generated document of at least one document template contains more components than the document template for at least one document request.

85. (currently amended) The software development system as in claim 74, wherein multiple instances of at least one third component denoted on the document template can be included in the generated document.

86. (currently amended) The software development system as in claim ~~85~~74, further comprising seventh instructions to assign a unique identifier to each component

APPENDIX

instance of at least one seventh component, whereby the ~~third~~ seventh component includes eighth instructions to qualify names generated into the browser code with the unique identifier.

87. (currently amended) A software development system as in claim 74, wherein at least one ~~document template has a fourth and a fifth component denoted thereon in a way that the fourth component contains the fifth component, the fourth component containing~~ contains ninth instructions to decide about how many instances of the ~~fifth components nested inside the fourth component~~ are included into the generated document.

88. (currently amended) A software development system as in claim 74 the editor able to provide an editable view that includes multiple instances of selected components similarly as the final application.

89. (previously amended) A software development system as in claim 74 wherein at least one sixth component includes tenth instructions to display the sixth component, the tenth instructions being used to generate browser code for displaying the sixth component during editing as well as during normal use of the component.

90. (currently amended) An editor for use with a web browser, the editor allowing the user to edit at least one document displayed by the browser, wherein ~~clicking on said document displayed in the browser initiates editing functions, and scripts contained in said document remain functional during editing, the editor including a first software program for execution within the browser and for processing the clicking~~ selected clicks on the view of said document displayed in the browser by initiating editing functions .

91. (previously added) The editor as in claim 90 using at least two windows, a first browser window displaying said document and a second window for displaying information on an element contained in said document.

APPENDIX

92. (currently amended) The editor in claim 90 further comprising a second software program for modifying said documents in cooperation with the first software program.

93. (currently amended) The editor as in claim ~~90~~ 92 further comprising a third program for transforming the document into a generated document ~~thereby adding editing features,~~ the browser displaying the generated document as said view looking similar to the original and interpreting ~~the~~ editing features contained in the generated document.

94. (currently amended) The editor as in claim 93 wherein said document is a dynamic document having components denoted thereon, the third software program further comprising instructions for generating browser code in cooperation with selected components.

95. (previously added) The editor as in claim 94 wherein the browser together with the first software program is running on a client computer connected to a server computer via a data network, wherein the second and the third software program run on the server computer.

96. (previously added) The editor in claim 90 wherein links contained in said document stay functional allowing the user to browse and edit at the same time.

97. (previously amended) A system for displaying dynamically generated documents in a data network coupling a server computer to a client computer, wherein the client computer has a first software program including first instructions for generating a document request to obtain at least one generated document from the server computer and for displaying the generated document, comprising:

a plurality of components for execution on the server computer, including a first component including second program instructions to generate browser code and third

APPENDIX

program instructions for execution on the server which are initiated by the user interacting with the first component, and,

fourth program instructions on the server computer for, based on data contained in a document request initiated by the first software program on the client computer, generating generated documents for transfer to the client computer and display by the first software program, thereby calling second program instructions of components.

98. (previously added) The system of claim 97, the server computer further comprising fifth program instructions for analyzing said data and for calling third program instructions of first component as necessary.

99. (previously added) The system of claim 98, further comprising a plurality of document templates residing in the data store, at least one of the document templates having at least one second component denoted thereon.

100. (previously added) The system of claim 99, wherein tag syntax is used to denote the second component, whereby the tag name identifies a component.

101. (previously added) The system of claim 99, wherein at least one third component denoted on a document template contains the first component.

102. (previously added) The system of claim 101, wherein third components implementation scheme includes logic to decide how often to insert the first component into the generated document.

103. (previously added) The system of claim 98, the server computer further comprising sixth program instructions for, based on the document request, deciding to insert more than one instance of the first component into the generated document.

APPENDIX

104. (previously added) The system of claim 103, making sure that multiple instances of the first component do not interfere by qualifying names generated into the browser code using unique identifiers.

105. (previously added) The system of claim 98, the server computer further comprising seventh program instructions for, based on the data, deciding to exclude the first component from the generated document.

106. (previously amended) The system of claim 105, wherein fifth instructions call third instructions only if the first component was contained on a page previously transferred to the client.

107. (previously added) The system of claim 98, wherein fifth program instructions include eighth program instructions to analyze said data for user interactions with multiple components and to call third program instructions of multiple components as necessary.

108. (previously added) The system of claim 107, wherein components include ninth instructions to check for errors and fifth instructions include tenth instructions to call ninth instructions of components as necessary and to suppress subsequent calling of third instructions in case of errors.

109. (previously added) The system of claim 98, further comprising eleventh program instructions for storing a data object in session memory representing at least one component instance included in a generated document, said eleventh program instructions for execution while dynamically generating a document.

110. (previously added) The system of claim 109, wherein third program instructions are encapsulated in a method of data objects, fifth program instructions including twelfth program instructions for identifying the data object that represents a

APPENDIX

component instance the user interacted with and for calling said method of said data object as necessary.

111. (previously added) The system of claim 110, further including thirteenth instructions for deciding based on said data to include more than one instance of a component into the generated document.

112. (previously added) The system of claim 109, further comprising twelfth program instructions for assigning a unique identifier to the first component instance, for associating the unique identifier with the data object, and for including the unique identifier into the generated document, said twelfth program instructions for execution while dynamically generating a document .

113. (previously added) The system of claim 112, wherein fifth program instructions analyze said data for unique identifiers and include thirteenth instructions for identifying the associated data object.

114. (currently amended) A system for displaying dynamically generated documents in a data network coupling a server computer to a client computer, wherein the client computer has a first software program including first program instructions for generating a request to obtain at least one generated document from the server computer and for displaying the generated document, comprising:

a plurality of components for execution on the server, at least one of the components including first features to cooperate with an editor in editing said component and second program instructions to generate browser code, and

third program instructions on the server for, based on the data contained in a request initiated by the client computer, generating generated documents for transfer to the client computer, thereby calling second program instructions of selected components.

115. (previously added) The system of claim 114 wherein first features include fourth program instructions for passing information to the editor.

APPENDIX

116. (currently amended) The system of claim 115 wherein at least part of said information is collected during execution of ~~the~~ selected components on the server.

117. (previously added) The system of claim 115 wherein said information is transmitted from the server to the client.

118. (currently amended) The system of claim 115 wherein said information includes attributes values of said component.

119. (previously added) The system of claim 114 wherein first features include fifth instructions that display additional editing features of the component during editing.

120. (previously added) The system of claim 119 wherein said editing features include handles.

121. (currently amended) The system of claim 114 wherein first features include an extension for use by the editor, said extension for enabling editing of an attribute value of the components ~~attributes~~.

122. (currently amended) The system of claim 121 wherein said extension is enables display of a page for editing the components attributes values.

123. (currently amended) The system of claim 114 wherein at least one components ~~is are~~ denoted on at least one document templates using tag syntax, whereby the tag name identifies a component kind.

124. (previously amended) The system of claim 114 containing at least one component wherein second program instructions are used to generate browser code for displaying the component during editing and during normal use.

APPENDIX

125. (currently amended) A method for editing an application that is built using components and that operates by generating documents comprising the steps of:

running the application, thereby executing selected components and generating a generated document,

displaying a view of the generated document,

selecting a component by clicking on selected portions of said view,

identifying the selected component in the source code of the application,

initiating a modification function modifying the source code of the application.

126. (previously added) The method of claim 125 wherein the running step and the displaying step are repeated after applying a modification function.

127. (previously added) The method of claim 125 further comprising collecting edit information for use by the identifying step.